

Further Results on Languages of Membrane Structures

Rama Raghavan

Department of Mathematics
Indian Institute of Technology Chennai
Chennai, India
ramar@iitm.ac.in

H. Ramesh

Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati, India
ramesh_h@iitg.ernet.in

Marian Gheorghe

Department of Computer Science
The University of Sheffield
Sheffield, UK
M.Gheorghe@dcs.shef.ac.uk

Shankara Narayanan Krishna

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai, India
krishnas@iitb.ac.in

In [3], P systems with active membranes were used to generate languages, in the sense of languages associated with the structure of membrane systems. Here, we analyze the power of P systems with membrane creation and dissolution restricted to elementary membranes, P systems without membrane dissolution operating according to certain output modes. This leads us to characterizations of recursively enumerable languages.

1 Introduction

In [3], an alternative approach to generate languages by means of P systems was proposed. An appropriate representation for a string was built by means of a membrane structure and then the string is generated by visiting the membrane structure according to a well-specified strategy. P systems with active membranes were considered, allowing membrane creation or division or duplication and dissolution, where the output of a computation may be obtained either by visiting the tree associated with the membrane structure, or by following the traces of a specific object, called traveller, or sending out the objects. For each of these approaches, characterizations of recursively enumerable languages were provided based on P systems that use different sets of operations for modifying the membrane structure.

The output of a computation was considered not as a single entity, which is either sent out of the system or collected in a specific membrane. Instead the output is given by concatenating the content or the labels of each region of the whole configuration reached by the system at the end of a computation. They considered a general class of P systems with active membranes equipped with membrane division, creation, duplication and dissolution operations. Membrane duplication means that, starting from an existing membrane, we can create a new membrane which encloses the existing one. Then three different approaches for collecting the output of a computation was given namely visiting the tree associated with the membrane structure, following the traces of a special object (traveller traces), and sending out the objects (external mode). The trace mode and external mode were investigated earlier in the literature. For the external mode, the main difference with respect to this approach is that, before sending out the objects, we need to prepare an appropriate membrane structure where the output objects are supposed to be distributed according to a specific strategy.

The approach presented in [3], is related to the problem of finding alternative ways to define the output of the computation in membrane systems. In fact, this method puts emphasis on the structure of the membranes whose role is important in successful computations.

In this paper, we investigate the computational power of P systems with active membranes equipped with membrane creation and membrane dissolution restricted to elementary membranes operating according to all four output modes. Also we analyse the power of P systems without membrane dissolution operating according to the three identified output modes. We need the label changing feature of *in* type rules to obtain the universality in the second case.

The paper is organized as follows. Section 3 recalls the definition of P systems with active membranes together with the definition of three different output modes. In section 3.1 we state the results from [3] concerning the power of P systems with active membranes generating languages of membrane structures. In sections 4 and 5, we prove characterizations of recursively enumerable languages by means of P systems with active membranes equipped with the membrane creation and dissolution operations.

2 Some Prerequisites

In this section we introduce some formal language theory notions which will be used in this paper; for further details, refer to [8].

For an alphabet V , we denote by V^* the set of all strings over V , including the empty one, denoted by λ . By *RE* we denote the family of recursively enumerable languages.

In our proofs in the following sections we need the notion of a *matrix grammar with appearance checking*. Such a grammar is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n), n \geq 1$, of context free rules over $N \cup T$, and F is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$ we write $w \Rightarrow z$ if there is a matrix $(A_1 \rightarrow x_1, \dots, A_n \rightarrow x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, are such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either (1) $w_i = w'_i A w''_i$, $w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or (2) $w_i = w_{i+1}, A_i$ does not appear in w_i , and the rule $A_i \rightarrow x_i$ appears in F . (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied - one says that these rules are applied in the *appearance checking mode*).

The language generated by G is defined by $L(G) = \{w | w \in T^*, S \Rightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac} . It is known that $MAT_{ac} = RE$.

We say that a matrix grammar with appearance checking $G = (N, T, S, M, F)$ is in the *Z-binary normal form* if $N = N_1 \cup N_2 \cup \{S, Z, \#\}$, with these sets mutually disjoint, the matrices of type 3 can also be of the form $(X \rightarrow Z, A \rightarrow \#)$, and the only matrix of type 4 (terminal matrix) is of the form $(Z \rightarrow \lambda)$.

According to Lemma 1.3.7 in [4], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

Next we define a computing device which is equivalent in power with Turing machine. Such a machine runs a program consisting of numbered instructions of several simple types. Several variants of register machines with different number of registers and different instruction sets were shown to be computationally universal (e.g., see [5]).

An *n-register machine* is a construct $M = (n, H, l_0, l_h, I)$, where:

- n is the number of registers,
- H is the set of instruction labels,
- l_0 is the initial label,
- l_h is the final label, and

- I is a set of labelled instructions of the form $l_i : (op(r), l_j, l_k)$, where $op(r)$ is an operation on register r of M , l_i, l_j, l_k are labels from the set H (which labels the instructions in a one-to-one manner),

The machine is capable of the following instructions:

$(ADD(r), l_j, l_k)$: Add one to the contents of register r and proceed to instruction l_j or to instruction l_k ; in the deterministic variants usually considered in the literature we demand $l_j = l_k$.

$(SUB(r), l_j, l_k)$: If register r is not empty, then subtract one from its content and go to instruction l_j , otherwise proceed to instruction l_k .

halt: Stop the machine. This additional instruction can only be assigned to the final label l_h .

When considering the generation of languages, we use the model of a *register machine with output tape* (e.g., see [2]), which also uses a tape operation:

- $l : (write(a), l'')$: Write symbol a on the the output tape and go to l'' .

We then also specify the output alphabet T in the description of the register machine with output tape, i.e., we write $M = (n, T, H, l_0, l_h, I)$. Let $L \subseteq V^*$ be a recursively enumerable language. Then L can be generated by a register machine with output tape and with 2 registers.

3 Languages of Membrane Structures

We consider a general class of P systems with active membranes equipped with membrane division, creation, duplication and dissolution operation. These operations represent abstractions of cellular biology processes of mitosis and membrane formation through self-assembling lipid bilayers [1]. We recall the definition of this P system from [3].

Definition 1 *A P system with active membranes is a construct*

$$\Pi = (V, K \cup \{0\}, \mu, w_0, w_1, \dots, w_{m-1}, R)$$

where

1. V is an alphabet; its elements are called objects;
2. K is an alphabet; its elements are called labels; the symbol $0 \notin K$ is the label of the skin membrane;
3. μ is a membrane structure containing $m \geq 1$ membranes; the skin membrane is labelled by 0 and all other membranes are labelled with symbols in K ;
4. w_0 is the multiset associated with the skin membrane;
5. $w_i \in V^*$, for $1 \leq i \leq m-1$, is a multiset of objects associated with the membrane i ;
6. R is a finite set of rules of the form:
 - a) $[i a \rightarrow v]_i$ with $a \in V, v \in V^*$, and $i \in K \cup \{0\}$ (inside a membrane i an object a is replaced by a multiset v),
 - b) $[i a]_i \rightarrow b[i]_i$ with $a, b \in V$, and $i \in K \cup \{0\}$ (an object is sent out from a membrane, maybe modified),
 - c) $a[i]_i \rightarrow [i b]_i$ with $a, b \in V$, and $i \in K \cup \{0\}$ (an object is moved into a membrane, the object may be modified),

- d) $[_i a \rightarrow [_j b]_j]_i$ with $a, b \in V$, $i \in K \cup \{0\}$, and $j \in K$ (membrane creation: inside a membrane i , starting from an object a , a new elementary membrane j is created, which contains an object b),
- e) $[_i a]_i \rightarrow [_k b]_k [_j c]_j$ with $a, b, c \in V$, $i, j, k \in K$ (membrane division: the membrane i , in the presence of an object a , is divided into two new membranes labelled by k and j , and the content (objects and sub-membranes) of the membrane i is copied into each new membrane where the object a is respectively replaced by b or c),
- f) $[_i a]_i \rightarrow [_k b [_j c]_j]_k$ with $a, b \in V$, $i, j, k \in K$ (membrane duplication the membrane i , in the presence of an object a , is duplicated, that is, the label i is changed into j , the object a is replaced by c , and a new upper membrane labelled by k is created, which contains an object b),
- g) $[_i a]_i \rightarrow a$ with $a \in V$, and $i \in K$ (membrane dissolution: in the presence of an object a , the membrane i is dissolved and its content (objects and sub-membranes) is released in the directly upper region).

In the above system we have: an initial membrane structure with m membranes that contain m multi-sets associated with the regions, and a finite set of evolution rules. Moreover, as usual in P systems with active membranes, we also consider a distinct alphabet K which is used to label the membranes and is necessary to precisely identify the rules that can be applied inside every membrane. In general, in a P system with active membranes, the number of membranes can be increased and decreased arbitrarily and there can be many different membranes with the same label, which can be distinguished from each other only by the objects they contain. Thus, the labels from K make possible to keep finite the representation of a P systems by specifying a set of "types", each one with its own set of rules, for the membrane possibly present in the system at any time. A membrane with no further membrane inside is called an *elementary membrane*.

The set R contains rules for modifying both the number and the distribution of objects inside the system and the number and type of membranes which define the structure of the system. The former rules are expressed in the form of transformation and communication rules (rules of type (a), (b) and (c)) whereas the latter ones (rules of type (d), (e), (f) and (g)) comprise the operations of: membrane creation, membrane division, membrane duplication and membrane dissolution, respectively.

Remark 1 Here, we do not consider the feature of membrane polarization for P systems with active membranes as reported in the literature. However, in the above definition, rules of more general forms are used that are able to change the labels of the membranes involved.

As usual, P systems with active membranes evolve according to a non-deterministic maximal parallel strategy. Rules of type (a), (b), (c), and (d) are applied to all the objects which they can be applied to. Rules of type (e), (f), (g) are applied to all the membranes which they can be applied to. Obviously, in each step, the same membrane cannot be used by more than one rule of type (e), (f), (g) (i.e., a membrane cannot simultaneously be divided, duplicated and dissolved). More precisely, we assume that, in each step, the objects first evolve by means of rules of type (a), (b), (c), (d), and then the membranes evolve according to rules of type (e), (f), (g).

A computation is obtained by applying rules of R starting from the initial configuration. A computation is said successful if it reaches a configuration where no more rules can be applied.

We illustrate the application of rules (d) - (g) by examples. In the following examples, P, Q are possible contents of membranes and a, b, c are objects from V . The effect of the rules on some membrane structures is below:

- (1) $[_i P a]_i$
rule of type (d) : We get $[_i P [_j b]_j]$
rule of type (e) : We get $[_j P b]_j [_k P c]_k$
rule of type (f) : We get $[_k b [_j P c]_j]_k$
- (2) $[_j [_i P a]_i Q]_j$
rule of type (g) : We get $[_j P Q a]_j$

The result of a computation may be considered in various forms, which are called output modes.

- *Visiting the tree.* The result of a computation is the set of strings obtained by visiting the tree associated with the membrane structure in the final configuration. The resulting set of strings is obtained by concatenating either the labels of the membranes or the objects inside these membranes, in the order they are visited. If a membrane contains more than one object, then we consider all the possible permutations of these objects. When we collect the labels, we do not consider the skin membrane, which is always labelled by 0. This output mode is denoted either by *lab*, if we collect the labels, or by *obj* if we collect the objects.
- *Traveller traces.* We assume that the initial configuration contains a special object t , called the *traveller*, inside some membrane. The traveller t can be moved by using rules of type (b) or (c), but it cannot be modified by any rule. The resulting string is obtained as follows: initially we start with the empty string associated with the initial configuration, then whenever the object t crosses a membrane labelled by i , we add the symbol i at the rightmost side of the current string. This output mode is denoted by *traces*.
- *External mode.* The resulting set of strings is defined as follows: we start initially with an empty string outside of the membrane system; whenever an object is sent out of the skin membrane, we add such an object to the rightmost end of each current string. If some objects are sent out from the skin membrane at the same time, we consider the string formed by all the permutations of these objects. This output mode is denoted by *ext*.

We denote by $LOP_{m,n}(Op, l)$, with $Op \subseteq \{a, b, c, d, e, f, g\}$, $l \in \{obj, lab, traces, ext\}$, the family of languages generated by P systems with active membranes with at most m membranes in the initial configuration that use at most n different labels for the membranes (the cardinality of $K \cup \{0\}$ is at most n), that apply rules of the forms specified in Op , and has the output mode l . As usual, if the value of m , or the value of n , is not bounded, it is replaced by the symbol $*$. Moreover, when the rules of type (e), (f), and (g) are allowed only for elementary membranes, the corresponding operations are denoted by e' , f' , and g' . Also, when the rules of type (b) and (c) are allowed to change the label of the membrane, the corresponding operations are denoted by b' and c' .

3.1 The power of membrane creation and membrane division

In this section, we present some results from [3]. The universality of P systems with membrane division and membrane dissolution with respect to the output modes *lab*, *obj* was given by the following theorem.

Theorem 1 $LOP_{1,*}(\{a, b, c, d, g\}, v) = RE$, for $v \in \{lab, obj\}$.

A similar result holds for P systems with membrane division and membrane dissolution restricted to elementary membranes.

Theorem 2 $LOP_{2,*}(\{a, b, c, e, g'\}, v) = RE$, for $v \in \{lab, obj\}$.

In the proofs of the above theorem, the final configuration has membrane structure of depth 2. So a “predefined” order among the membranes is necessary to get a suitable representation for the strings of a language. Such an approach does not work well in the case of *traces* and *external* output modes. The following theorem shows that such a problem can be avoided by considering the operation of *membrane duplication*.

Theorem 3 $LOP_{2,*}(\{a,b,c,f,g\},v) = RE$, for $v \in \{lab,obj,ext\}$ and $LOP_{3,*}(\{a,b,c,f,g\},traces) = RE$.

The following cases were left open in [3]:
The computational power of

1. P systems with membrane creation and membrane dissolution (or membrane division and membrane dissolution) operating according to the external mode or the traces mode;
2. P systems without membrane dissolution operating according to any of the three identified output modes;
3. P systems with membrane creation and membrane dissolution restricted to elementary membranes.

We settle some of the above cases in the coming sections.

4 Universality with Membrane Creation and Dissolution

The following is a characterization of recursively enumerable languages by means of P systems with active membranes equipped with the membrane creation and dissolution operation restricted to elementary membranes operating according to all output modes.

Theorem 4 $LOP_{1,*}(\{a,b,c,d,g'\},v) = RE$, for $v \in \{lab,obj,ext\}$ and $LOP_{2,*}(\{a,b,c,d,g'\},traces) = RE$.

Proof: The proof is based on the simulation of a register machine $M = (2, T, P, l_0, l_h)$. We construct a P system with active membranes that simulates the register machine M such that

$$\Pi = (V, K \cup \{0\}, [0]_0, l'_0, R)$$

where

$$\begin{aligned} V &= T \cup \{a_1, a_2, b_1, b_2\} \cup \{l', l'' \mid l : (ADD(r), l', l'') \in P\} \\ &\cup \{l_0, l'_0, l''_0, 1', 2', \$' \mid l_0 \text{ is the initial label of } M\} \\ &\cup \{l_i, l'_i, l''_i, l'''_i, l''''_i, l', l'' \mid l : (SUB(i), l', l'', i = 1, 2)\} \\ &\cup \{(a, l'), (\overline{a, l'}) \mid l : (WRITE(a), l') \in P, a \in T\} \\ &\cup \{1', 2', \$'\} \\ K &= T \cup \{1, 2, 3, 4, \$\} \end{aligned}$$

$$\begin{aligned}
R = & \{[0l'_0 \rightarrow 1'2'\$l''_0]_0, [0l''_0 \rightarrow l_0] \mid l_0 \text{ is the initial label of } M\} \\
\cup & \{[01' \rightarrow [1]_1]_0, [02' \rightarrow [2]_2]_0, [0\$' \rightarrow [\$]_\$]_0\} \\
\cup & \{[0l \rightarrow b_i l']_0, [0l \rightarrow b_i l'']_0 \mid l : (ADD(i), l', l''), i = 1, 2\} \\
\cup & \{b_i [i]_i \rightarrow [i a_i]_i \mid i = 1, 2\} \\
\cup & \{l \rightarrow l_i, l_i [i]_i \rightarrow [i l_i]_i \mid l : (SUB(i), l', l''), i = 1, 2\} \\
\cup & \{a_i [3]_3 \rightarrow [3 a_i]_3, [3 l'_i \rightarrow l''_i]_3 \mid l : (SUB(i), l', l''), i = 1, 2\} \\
\cup & \{[3 a_i]_3 \rightarrow \lambda, [3 l''_i \rightarrow l'''_i]_3, [3 l'''_i]_3 \rightarrow l_i^{iv} \mid l : (SUB(i), l', l''), i = 1, 2\} \\
\cup & \{[i l_i''']_i \rightarrow l', [i l_i^{iv}]_i \rightarrow l'' \mid l : (SUB(i), l', l''), i = 1, 2\} \\
\cup & \{[0l \rightarrow (a, l')]_0 \mid l : (WRITE(a), l')\} \\
\cup & \{(a, l') [b]_b \rightarrow [b(a, l')]_b \mid b \in T \cup \{\$\}\} \\
\cup & \{[\$(a, l')]_\$ \rightarrow \overline{(a, l')}, [b \overline{(a, l')}] \rightarrow [a l' \$]_a]_b \mid b \in T\} \\
\cup & \{[a \$ \rightarrow [\$]_\$]_a \mid a \in T\} \\
\cup & \{l_h [1]_1 \rightarrow [1 l_h]_1, [1 l_h]_1 \rightarrow l'_h, l'_h [2]_2 \rightarrow [2 l'_h]_2, [2 l'_h]_2 \rightarrow l''_h\} \\
\cup & \{[0 a_i \rightarrow \lambda]_0 \mid i = 1, 2\} \\
\cup & \{l''_h [4]_4 \rightarrow [4 l''_h]_4, [4 l''_h]_4 \rightarrow \lambda\}
\end{aligned}$$

Let us see how the P system Π works. Initially, we have the configuration $[0[4t]4l'_0]_0$. We apply the first 5 rules to produce the configuration $[0l_0[1]_1[2]_2[\$]_\$]_0$. The value of the two registers $i = 1, 2$, are represented by the number of objects a_i inside the corresponding membrane i . The membrane labelled $\$$ is used to prepare an appropriate membrane structure where the output objects are supposed to be distributed according to a specific strategy.

The add instruction $l : (ADD(i), l', l'')$ is simulated as follows. We use the rule $l \rightarrow b_i l'$ or $l \rightarrow b_i l''$ to create an object b_i corresponding to the register i . Now the object b_i changes to a_i while entering inside membrane i .

In order to simulate a subtract instruction $l : (SUB(i), l', l'')$, we send the object l_i into the membrane i and then proceed in the following way: The object l_i creates a membrane with label 3 and an object l'_i . If the register i is not empty, then the object a_i will enter membrane 3 and dissolve it; otherwise the object l_i''' dissolves membrane 3 there by changing to l_i^{iv} . If the register i is not empty, then we have l_i''' in membrane i ; otherwise l_i^{iv} . Now we will send l' or l'' to the skin membrane depending upon the presence of l_i''' or l_i^{iv} in membrane i respectively. This will end the simulation of the SUB instruction.

The simulation of the instruction $l : (WRITE(a), l')$ is done as follows. First we use the rule $l \rightarrow (a, l')$ in the skin membrane. The object (a, l') travels deep inside the nested membrane structure until it reaches the membrane $[\$]_\$$. In membrane $\$$, the object (a, l') changes to $\overline{(a, l')}$ and dissolves the membrane. Now the object $\overline{(a, l')}$ will create a membrane labelled a which contains the objects l' and $\$$. The object l' moves toward the skin membrane whereas the object $\$$ will create a membrane $[\$]_\$$ inside the membrane $[a]_a$. The object l' starts the simulation of the instruction labelled l' after reaching the skin membrane.

The presence of object l_h in the skin membrane will start the clean-up process. It will remove both the membranes 1, 2 and the objects inside them. Finally the object l''_h dissolves membrane 4 which contains the object t .

At last, we have a configuration of the form $[0t[x_1[x_2 \dots [x_h[\$]_{x_h} \dots]_{x_2}]_{x_1}]_0$ with $x_1 x_2 \dots x_h \in L(M)$, for some $h \geq 1$. Now we move the traveller t by using rules of the form $t[a]_a \rightarrow [at]_a$, with $a \in T$, and in this way we generates exactly the string $x_1 x_2 \dots x_h \in L(M)$.

External mode: For this mode we consider a P systems Π whose initial configuration is $[_0 l'_0]_0$, where l_0 is the starting label of the register machine M . We simulate the register machine M in the same way as described above for the traveller traces, and during the clean-up process the object l'_h changes to f and dissolves membrane 2 instead of changing to l''_h . Thus, we have a configuration $[_0 f[x_1[x_2 \dots [x_h[\$]\$]_{x_h} \dots]_{x_2}]_{x_1}]_0$, and we can generate the string $x_1 x_2 \dots x_h \in L(M)$ by using the following rules:

- $f[a]_a \rightarrow [a f']_a$ with $a \in T$
- $[a f']_a \rightarrow a f]_a$ with $a \in T$
- $[ba]_b \rightarrow a[b]_b$ with $a, b \in T$
- $[0a]_0 \rightarrow a[0]_0$ with $a \in T$

By applying these rules, we can send the objects out of the skin membrane in the right order

We can easily modify the above system Π to obtain a final configuration of the form $[_0 [x_1 x_1 [x_2 x_2 \dots [x_h x_h [\$]\$]_{x_h} \dots]_{x_2}]_{x_1}]_0$ for some $h \geq 1$, and $x_1 x_2 \dots x_h \in L(M)$. If we visit the tree associated with this membrane structure either by collecting the labels or by collecting the objects, then we get $x_1 x_2 \dots x_h \in L(M)$ in both cases. \square

Remark 2 *The universality of P systems with membrane division and membrane dissolution restricted to elementary membranes with respect to the traces and external output modes can be proved in a similar fashion provided the rules of type endocytosis were allowed. Because a combination of rules of type division and endocytosis can simulate rules of type creation.*

5 Universality with only Membrane Creation

A similar result holds for P systems that use only the membrane creation operation avoiding the operation membrane dissolution for all output modes except *traveller traces*. But we need the label changing feature for *in* type rules to obtain universality.

Theorem 5 $LOP_{1,*}(\{a, b, c', d\}, v) = RE$, for $v \in \{lab, obj, ext\}$.

Proof: Let $G = (N, T, S, M, F)$, with $N = N_1 \cup N_2 \cup \{S, Z, \#\}$, be a matrix grammar with appearance checking in Z-binary normal form where the matrix of type 1 is $(S \rightarrow XA)$, the matrices of type 2 are labelled, in one to one manner, by m_1, \dots, m_k , and matrices of type 3 by m_{k+1}, \dots, m_n . We construct a P system with active membranes that simulates the matrix grammar G as follows:

$$\Pi = (V, K \cup \{0\}, [0]_0, S, R)$$

where

$$\begin{aligned} V &= N_1 \cup N_2 \cup T \cup \{Z, \#\} \cup \{Y_i, Y'_i \mid 1 \leq i \leq n, Y \in N_1\} \\ &\cup \{Y_{i,B} \mid Y \in N_1, B \in N_2, 1 \leq i \leq k\} \cup \{Y_{i,\$} \mid Y \in N_1, 1 \leq i \leq k\} \\ K &= N_2 \cup T \cup \{\$\} \end{aligned}$$

$$\begin{aligned}
R = & \{ [{}_0S \rightarrow [{}_AX_A]_A]_0, [{}_AX_A \rightarrow [{}_SX'_A]_S]_A \mid (S \rightarrow XA) \in M \} \\
& \cup \{ [{}_SX'_A]_S \rightarrow X'_A[{}_S]_S, [{}_AX'_A]_A \rightarrow X[{}_A]_A \mid (S \rightarrow XA) \in M \} \\
& \cup \{ [{}_0X \rightarrow Y_i]_0 \mid X, Y \in N_1, 1 \leq i \leq n \} \\
& \cup \{ Y_i[y]_y \rightarrow [{}_yY_i]_y \mid y \in N_2 \cup T, y \neq A, m_i : (X \rightarrow Y, A \rightarrow x) \in M, 1 \leq i \leq k \} \\
& \cup \{ Y_i[{}_S]_S \rightarrow [{}_S\#]_\# \mid Y \in N_1, 1 \leq i \leq k \} \\
& \cup \{ Y_i[{}_A]_A \rightarrow [{}_aY'_i]_a \mid m_i : (X \rightarrow Y, A \rightarrow a), 1 \leq i \leq k \} \\
& \cup \{ [{}_yY'_i]_y \rightarrow Y'_i[{}_y]_y \mid y \in N_2 \cup T \cup \{ \$ \}, 1 \leq i \leq n \} \\
& \cup \{ [{}_0Y'_i \rightarrow Y]_0 \mid Y \in N_1, 1 \leq i \leq n \} \\
& \cup \{ Y_i[{}_A]_A \rightarrow [{}_aY_{i,a_2}]_{a_1} \mid m_i : (X \rightarrow Y, A \rightarrow a_1a_2), 1 \leq i \leq k \} \\
& \cup \{ Y_{i,B}[{}_C]_C \rightarrow [{}_BY_{i,C}]_B \mid B, C \in N_2 \cup T \cup \{ \$ \}, 1 \leq i \leq k \} \\
& \cup \{ [{}_BY_{i,S} \rightarrow [{}_SY'_i]_S]_B \mid Y \in N_1, B \in N_2, 1 \leq i \leq k \} \\
& \cup \{ Y_i[{}_A]_A \rightarrow [{}_S\#]_\#, Y_i[{}_S]_S \rightarrow [{}_SY'_i]_S \mid m_i : (X \rightarrow Y, A \rightarrow \#), k+1 \leq i \leq n \} \\
& \cup \{ [{}_S\# \rightarrow \#]_\#, [{}_0Z \rightarrow \lambda]_0 \}
\end{aligned}$$

Initially, we have the configuration $[{}_0S]_0$. We simulate the unique matrix of type 1 in G by applying the first 4 rules to produce the configuration $[{}_0X[{}_A[{}_S]_S]_A]_0$. We need the membrane labelled by $\$$ in order to identify the end of the string.

Assume that we have a configuration of the form $[{}_0X[x_1[x_2 \dots [x_h[{}_S]_{x_h} \dots]_{x_2}]_{x_1}]_0$ after some steps, where $h \geq 1$, and $Xx_1x_2 \dots x_h$ is a sentential form of G , with $X \in N_1, x_i \in N_2 \cup T$. Now we apply the rule $[{}_0X \rightarrow Y_i]_0$, for some $1 \leq i \leq n$. We have two cases according to the values of i .

Case 1: $1 \leq i \leq k$. In this case, we are simulating a matrix of type 2, i.e., $m_i : (X \rightarrow Y, A \rightarrow x)$. By using the rule $Y_i[y]_y \rightarrow [{}_yY_i]_y$, we move Y_i deeper inside the nested membranes. If there is no membrane labelled by A in the current configuration, then we use the rule $Y_i[{}_S]_S \rightarrow [{}_S\#]_\#$. The symbol $\#$ generates an infinite computation by means of the rule $[{}_S\# \rightarrow \#]_\#$. If some membrane labelled A is present in the current configuration, then we have two cases.

Case a): $|x| = 1$, i.e., $x = a \in N_2 \cup T$.

In this case, we use the rule $Y_i[{}_A]_A \rightarrow [{}_aY'_i]_a$. The above rule changes the object Y_i into Y'_i and also changes the label A into a . Now the object Y'_i travel towards the skin membrane. Once it reaches the skin, it becomes Y .

Case b): $|x| = 2$, i.e., $x = a_1a_2 \in N_2 \cup T$.

In this case, we use the rule $Y_i[{}_A]_A \rightarrow [{}_aY_{i,a_2}]_{a_1}$. Here the object Y_i changes to Y_{i,a_2} and the label A changes to a_1 . Further we use the rule $Y_{i,B}[{}_C]_C \rightarrow [{}_BY_{i,C}]_B$ to move the object $Y_{i,B}$ towards the membrane labelled $\$$. While moving the objects $Y_{i,B}$, we change the labels of the membrane remembering the previous label in their second component. Once we got the object $Y_{i,S}$ in the innermost membrane, we use it to create a membrane labelled $\$$ containing the object Y'_i . After this we move Y'_i towards the skin membrane. It will become Y once it reaches the skin membrane.

Case 2: $k+1 \leq i \leq n$. That is we are simulating a matrix of type 3 ($m_i : (X \rightarrow Y, A \rightarrow \#)$). In this case, we use the object Y_i to check for the presence of a membrane labelled A in the current configuration. If there exists a membrane labelled A , the object Y_i is moved inside by the rule $Y_i[{}_A]_A \rightarrow [{}_S\#]_\#$. This will lead to an infinite computation that yields no result. Otherwise, the object Y_i becomes Y'_i after reaching the innermost membrane labelled by $\$$. Now we move the object Y'_i towards the skin membrane where it changes to Y .

Finally, we erase the symbol Z in the skin membrane, once it was introduced and the computation halts. Now by applying rules as in the previous theorem, we send out the objects in the right order. \square

6 Conclusion

This paper explores the idea of defining membrane systems that are able to build up a membrane structure that encodes some meaningful information proposed by [3]. We investigated the computational power of P systems with membrane creation and dissolution rules operating according to the external and traces mode. Also we proved the universality of P systems with membrane creation alone, but we allow the label changing feature for *in* type rules. At the moment, we are unable to characterize the power of P systems with active membranes equipped with membrane creation alone.

Acknowledgements. M. Gheorghe and R. Rama are grateful to the Royal Academy of Engineering which through a grant supporting *Exchanges with India and China* (2010), partially funded this research. MG has been also supported by CNCSIS grant no. 643/2009, *An integrated evolutionary approach to formal modelling and testing*. The authors would like to thank the anonymous reviewers for the comments made on an early version of this paper.

References

- [1] B. Alberts et al., *Molecular Biology of the Cell*, Garland Science, 2002.
- [2] A. Alhazov, R. Freund and A. Riscos-Núñez, One and Two Polarizations, Membrane Creation and Objects Complexity in P Systems, *Proc. of SYNASC 2005*, Timisoara, Romania, 385–394.
- [3] F. Bernardini and M. Gheorghe, Languages Generated by P Systems with Active Membranes, *New Generation Computing*, 22(4), 2004, 311–329.
- [4] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, 1989.
- [5] M.L. Minsky, *Computation: Finite and Infinite Machines*, Prentice Hall, 1967.
- [6] Gh. Păun, *Membrane Computing: An Introduction*, Springer-Verlag, Berlin, 2002.
- [7] R. Rama and H. Ramesh, On Generating Trees by P Systems with Active Membranes, *Proc. of SYNASC 2005*, Timisoara, Romania, 462–466.
- [8] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages* (3 volumes), Springer-Verlag, 1997.